PRINT ISSN 1119-8362
Electronic ISSN 1119-8362

Full-text Available Online at
https://www.ajol.info/index.php/jasem
http://ww.bioline.org.br/ja

J. Appl. Sci. Environ. Manage.
Vol. 25 (10) 1759-1764 October 2021

An Enhanced Performance Analysis of Software Using Architectural Feedback

***SALLA, NS; GARBA, M; DANLAMI, G**

Department of Computer Science, Kebbi State University of Science and Technology, Aliero, Kebbi State, Nigeria
**Corresponding Authors Email: sallastanley@gmail.com*
Other Authors Email: garbamga@gmail.com; gabsonley@gmail.com

ABSTRACT: The importance of software products and their quality attributes attainment has been a thing of concern in recent time to both academia and industry experts. This research work evaluated an enhanced performance analysis of software using architectural feedback. Data collected were, classified and analysed using SPSS reveal that the Relative Importance Index (RII) in relations to an enhance performance analysis of software using the architectural feedback was 0.83 which led to the proposal of a framework for an enhanced performance analysis of software using architectural feedback.

DOI: https://dx.doi.org/10.4314/jasem.v25i10.3

Copyright: *Copyright* © 2021 Salla *et al*. This is an open access article distributed under the Creative Commons Attribution License (CCL), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Dates: Received: 22 August 2021; Revised: 17 September 2021; Accepted: 06 October 2021

Keywords: Software, Performance analysis, Architectural Feedback, Performance Index, evaluation

With over dependence on software control systems and their effect on our everyday activities in every human endeavour, decisions concerning how these software artifacts are expected to interact with their environments are made by different experts in the software industry on regular basis. Some of these decisions deal with both functional and non-functional requirements. There exists a high interest in the early validation of performance requirements because it avoids late and expensive fix to consolidated software artifacts in the software development domain (Smith, 2005).

Current approaches to these problems are mostly based on the skills and experience of software developers or performance analysts. Software architecture means the structure of the system and it consists of software components, the externally visible properties of the components and the relationships among them (Elias and Jain 2010a). Making changes to the architecture in the later phases is difficult and complex. Architectural level analysis helps in identifying the potential problems in the early phase, when changes are not as complex and expensive to make (Obenza and Mendal 2014).

The quality of software can be considered from different quality attribute points of views (such as maintainability, modifiability, security and performance) (Elias G; Jain R, 2010b). Performance has always been problematic; this is attributed to the increase in size and complexity of today's software systems.

Predicting and guaranteeing performance before the system is built has become an interesting issue to both academia and industry expert in the software development domain in recent time. The validation of software architecture performance often finds obstacles to be accepted as a daily practice in software development processes for many reasons, one of the major reasons is lack of proper model to represent performance related quality attribute at the architectural state of the software to be developed and the developmental process or model deploy at the architectural level of the software development processes as little or no attention has been given to performance evaluation at the architectural state as an entity, numerous researchers has proposed different methods of looking at the other quality attributes as in Kazman et al., (2000a) proposed software architecture analysis method (SAAM) for analyzing modifiability of software products making use of scenario approach and their effects on the system, one of the greatest limitations of this method is that it relied on the experience of the analyst and that it offers a step-wise method for performing software architecture analysis with little or no scientific approach involved, and in 2008 they further proposed Architecture Tradeoff Analysis Method (ATAM) Kazman et al., (2000b) this approach has its own limitation as only the Tradeoff points are taken into consideration without minding

**Corresponding Authors Email: sallastanley@gmail.com*

how the tradeoff point can affect the performance of the entire system. In 2003, Dolan (2003), proposed Family Architecture Assessment Method (FAAM), this method emphasizes the strategic aspects that are associated with the evolutionary capabilities of the system. It only focuses on interoperability and extensibility of the software products with little or no concern for performance. While decisions made at every phase of the software development process can impact the quality of software product, architectural decisions have the greatest impact on quality attributes such as modifiability, reusability, reliability, and performance.

As Clements and Northrop note: "Whether or not a system will be able to exhibit its desired (or required) quality attributes is largely determined by the time the architecture is chosen." (Clements, P; Northrop, P, 2002).

This research work focused on an enhance analysis of software performance at the architectural level to ensure that they meet both functional and non-functional requirements highlighted by the stakeholders of the new software product and can provide a necessary output to the stakeholders as feedback before embarking on the development of these software product.

MATERIALS AND METHODS

Software architecture plays an important role in meeting a software system's performance. Performance depends largely on the frequency and nature of inter component communication and the performance characteristics of the components themselves.

The proposed system is expected to predict performance attributes using Mean, Mode and Standard Deviation, by transforming the specification of software architecture into desirable models. Then, timing information was added to this model. This method work based on availability of software artifacts, such as requirement and architecture specifications and design documents. Since performance is a runtime attribute, this method required suitable description of the dynamic behavior of a software system.

There are some challenges in architectural evaluation, including, for example: Software architectural evaluation requires an expert evaluation team to deal with unpredictable risks along with the known risks. For example, analyzing architectural decisions for satisfying security requirement is challenging. No one can accurately predict how often an attack will occur and how effectively security mechanisms will mitigate the damage. However, an experienced security manager in the evaluation team can estimate the risk and the effectiveness of proposed risk mitigation strategies.

There can be a lack of common understanding of the high level design. Software architects often do not document design rationale. When they do, they may not follow a systematic way of expressing the rationale.

There is also no standard notion to describe software architectures. However, currently Unified Modelling Language (UML) is widely used as an architectural specification language, but it is still not possible to express various architectural notations (e.g., quality attributes of interest).

As a result, quality attributes sometimes become vague for architectural analysis. The methodology used in this research was the Object-Oriented Analysis and Design (OOAD). OOAD is based on a set of building blocks which is an iterative software development process framework extensively using unified modelling language (UML) in the system development.

However, in this study, the researcher make used of Use-case and Class diagrams to illustrate the performance analysis of software using architectural feedback.

Data was collected with the aid of structured questionnaires that were comprehensive in content and very simple to understand.

The questionnaires was designed using Microsoft Google Forms which is an online tool due to the fact that it's easy, fast and can cover a wide range of respondents. Hence, the questionnaires was distributed and retrieved online via various online media i.e. WhatsApp, E-mail, etc.

Data presentation was done using tables in order to analyse the data collected effectively and efficiently for easy management and accuracy, likert scale, and Chi-square test was the major analytical tools used for this research.

And the result gotten from the analyzed data were formulated and transform into a class and use-case diagram.

THE USE CASE DIAGRAM OF THE SYSTEM

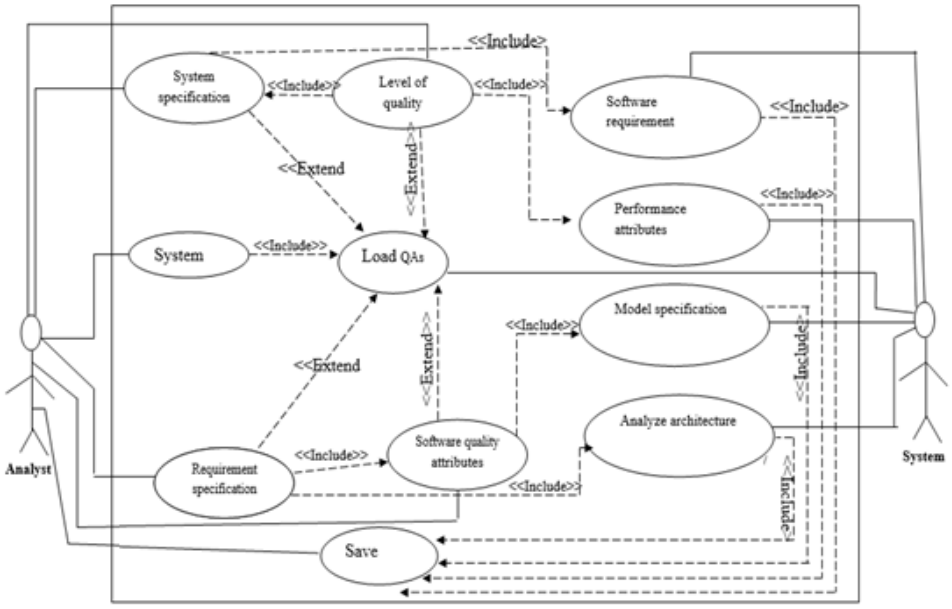

Fig 1. Use-Case diagram of the proposed scheme.

Class Diagram of our Propose Scheme.

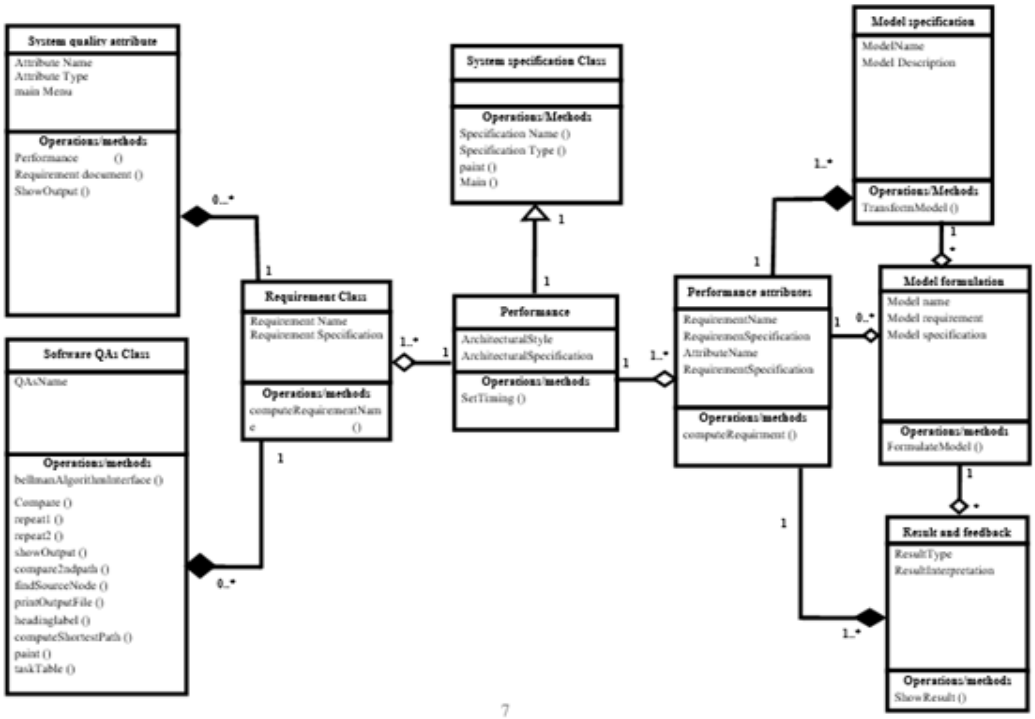

Fig 2. Class Diagram of the Propose Scheme

RESULTS AND DISCUSSION

With the results of our finding above we can now boldly formulate a framework that can serve as prototype using the results of our analysis as input in each stage of the architectural analysis of software that when duly adhere to can form our deliverables and expected outputs for an enhance performance analysis of software using architectural feedback.

Level 0 Data Flow Diagram: At the level zero (0) of our model applies that, the system analyst is an entity

that helps in processing and coordinating the activities determining the system requirement to specify the performance attributes of the system to be developed by reviewing the quality attribute specification as was seen on figure 5.

The system analyst with the stakeholders as seen in Table 1 and Table 2 generate the system requirement and specification at the architectural level by formulating a model that will aid in carrying out the requirement specification and analysing the quality attribute formulated at the specification stage of the architectural review by the stakeholders, analysing the result of the activities and sending a feedback to the analyst before proceeding to the development state of the system then the result stored in a database for future reference as can be seen in our level zero of the Data flow diagram of Table 1.

Table 2 represent stages where performance should be conducted at the architecture state of software performance analysis. The framework was formulated using Data Flow Diagram. A data flow diagram (DFD) illustrates how data is processed by a system in terms of inputs and outputs.

Table 1. Stage where performance analysis is conducted at the architectural level of software product development

	Mean ± Standard Deviation
At the initiation/Feasibility stage	3.00±0.62
At the Planning stage	3.31±0.46
At the Analysis stage	3.59±0.0
At the Design stage	3.11±0.83
At the Implementation stage	2.46±1.25
At the Maintenance stage	3.36±0.90
During project appraisal meeting	3.25±0.90
When project go live	2.68±1.03
Software analyst help in accommodating all requirements	3.53±1.03
Functionalities are generated from different stakeholders	3.42±0.50

Source: Researchers data, 2021.

Table 3. What to consider for involvement of stakeholders in requirement performance analysis

	Mean ± Standard Deviation
Provision of resources to the software architectural performance	3.17±0.38
Expertise and knowledge about the project	3.56±0.50
Company with competitive product	2.93±1.04
Financial contribution	3.25±0.44
Inclusion in similar project team	3.56±0.50
Authority over the project	3.34±0.68
Extensive usage of the developed system	3.30±0.46
Quality assurance standardization	3.04±1.09
At the initiation/feasibility stage	0.62±3.17

Source: Researchers data, 2021.

As the name indicates, its focus is on the flow of information, where data comes from, where it goes and how it gets stored. The data flow diagram (DFD) shows a functional perspective where each transformation represents a single function or process as indicated by the arrows.

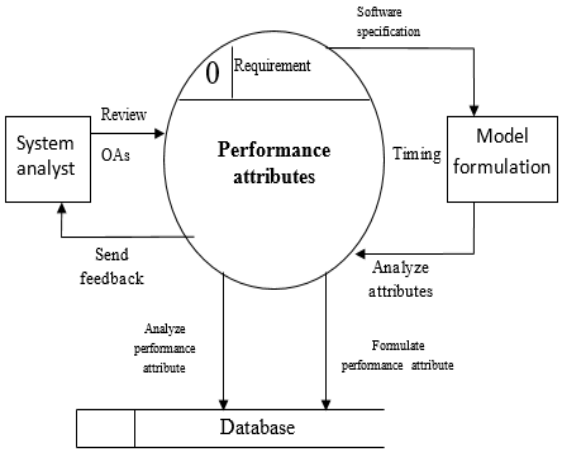

Fig 3. Level 0 Data Flow Diagram of our Proposed System

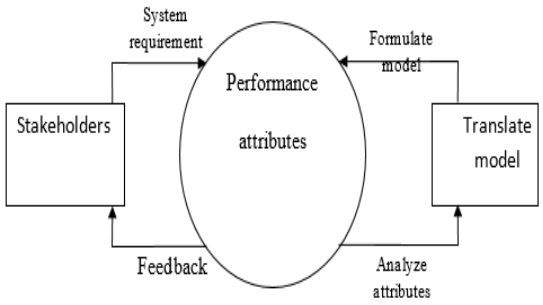

Fig 4. The context Flow Diagram of the proposed system

Context Flow Diagram of the proposed system: In the Figure 4 aboved, of our context flow diagram, the stakeholders are saddle with the responsibility of appointing the system analyst that will perform the system requirement elicitation of the necessary requirements of the software products, where the software quality attribute will be considered in this case performance, and a model for the system formulated to test the performance of the requirement specified, and translate the requirement into a software model to check for the suitability of the quality attributes using the requirement gathered by the requirements elicitation experts as seen in Figure 4 and figure 1, then the result of this phase of analysis should be send to the stakeholders as seen in Figure 4 before given directive for the design of the software products. If at any stage of the requirement analysis a phase is missed this whole process while have to start all over again.

Data Flow Diagram of the proposed System: The level 1 of our data flow diagram of our proposed framework,

further broke the level zero into different subunit and their different processes and role, at the beginning our entity the system analyst state all the performance attributes from all the requirement elicitated by the requirement elicitation team, from where a model will be form for the sole purpose of harnessing this performance attributes into a manageable state, by establishing the objective of the system and specifying the performance evaluation metrics that need to be conducted on the model by highlighting the quality attribute the model should consider and the nature of the prediction and properties to be return as attributes and the results should be evaluated after which the proposed feedback should be send to the system analyst to determine the suitability of the system and its functional capability before designing the system using the result obtained from this processes in Figure 5.

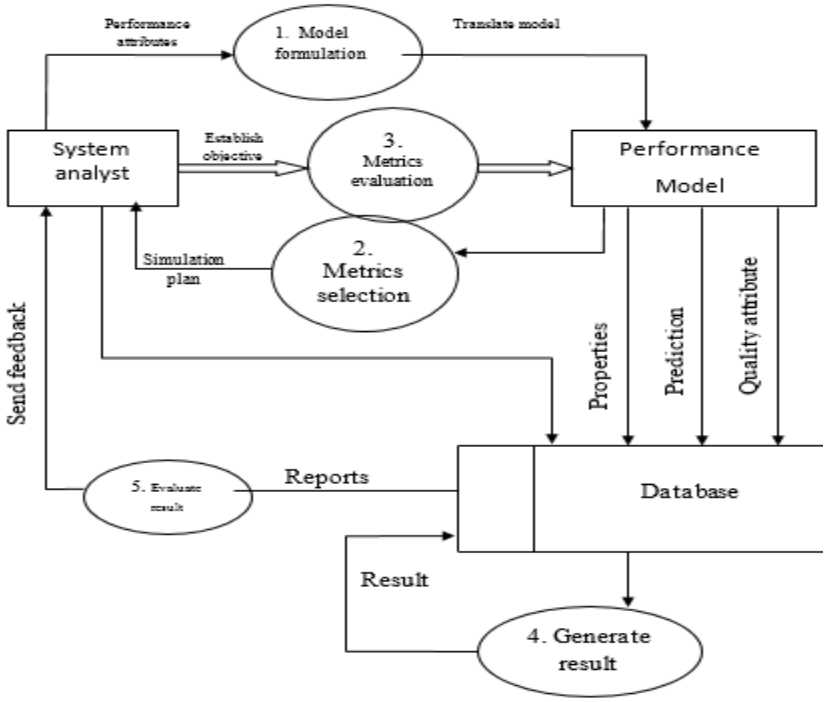

Fig 5. Level 1 Data Flow Diagram of the Proposed System

Conclusion: In this paper we have examined and explained the role software architectural analysis in the development of a software product, and some of the information required for this analysis and categories of people that should be involved in carrying out this analysis task. From data collected, analyzed and interpreted, vital results were obtained that was used to draw inference on who should be involved in software analysis and at what stage and a framework was also proposed and formulated that can aid in carrying out this task.

REFERENCES

Dolan, TJ (2003). Architecture assessment of information-system families: A practical perspective Clements, P; Northrop, L (2002). *Software product lines* (pp. 226-229). Boston: Addison-Wesley.

Elias, G; Jain, R (2010). Attributes Guided System Architecture Assessment. In *INCOSE Track at IEEE International Systems Conference 2010*.

Holt, R (2002). Software architecture as a shared mental model. *Proceedings of the ASERC Workhop on Software Architecture, University of Alberta*, 64.

Ionita, MT; Hammer, DK; Obbink, H (2002). Scenario-based software architecture evaluation methods: An overview. In *Workshop on methods and techniques for software architecture review and assessment at the international conference on software engineering* (pp. 19-24).

Jacobson, D; Stepanov, A; Chao, C; English, R; Wilkes, J (1992). *Mime: a high performance*

parallel storage device with strong recovery guarantees. Technical Report HPL-CSP-92-9.

Kazman, R; Carrière, SJ; Woods, SG (2000). Toward a discipline of scenario-based architectural engineering. *Annals of software Eng. 9*(1-2), 5-33.

Kazman, R; Klein M; Clements, P (2000). ATAM: A method for architecture evaluation. *Technical Report CMU/SEI-2000-TR-004*.

Obenza, R.; Mendal, G. (2014). Guaranteeing real-time performance using RMA. *Embedded Systems Programming*. *7*(5): 026-43.

Ramachandran, M; Chang, V (2016). Towards performance evaluation of cloud service providers for cloud data security. *Inter. J. Information Management*, *36*(4), 618-625.

Shah, MG (2014). Evaluation of the Software Architecture Styles from Maintainability Viewpoint. *Int. J. Comp. Sci. Info. Tech*. *6*(1).

Shanmugapriya, P; Suresh, RM (2012). Software architecture evaluation methods-a survey. *Inter. J. Comp. Applica*. *49*(16).

Smith CU. (2015). Performance Engineering of Software Systems. Addison-Wesley, Massachusetts. p570

Smith, C (2015), Performance Engineering of Software Systems. Addison-Wesley, Massachusetts. p570.

Smith, CU; Lladó, CM. (2009). Model interoperability for performance engineering: survey of milestones and evolution. In *International Workshop on Performance Evaluation of Computer and Communication Systems* (pp. 10-23). Springer, Be

Kazman, R; Klein M; Clements, P (2000). ATAM: A method for architecture evaluation. Technical Report CMU/SEI-2000-TR-004.

Obenza, R.; Mendal, G. (2014). Guaranteeing real-time performance using RMA. Embedded Systems Programming, 7(5): 026-43.

Ramachandran, M; Chang, V (2016). Towards performance evaluation of cloud service providers for cloud data security. International Journal of Information Management, 36(4), 618-625.

Shah, MG (2014). Evaluation of the Software Architecture Styles From Maintainability Viewpoint. int. Journal of computer science & information technology, 6(1).

Shanmugapriya, P; Suresh, RM (2012). Software architecture evaluation methods-a survey. International Journal of Computer Applications, 49(16).

Smith, CU. (2015). Performance Engineering of Software Systems. Addison-Wesley, Massachusetts. p570

Smith, C (2015), Performance Engineering of Software Systems. Addison-Wesley, Massachusetts. p570.

Smith, CU; Lladó, CM. (2009). Model interoperability for performance engineering: survey of milestones and evolution. In International Workshop on Performance Evaluation of Computer and Communication Systems (pp. 10-23). Springer, Berlin, Heidelberg